Towards personalized Keyword Search over Relational Databases

Haithem Ghorbel [1], Rim Faiz [2], Nouha Othman [3]

1. *Université de Tunis*
   LARODEC, Tunisia
   gho.haithem@gmail.com
2. *Université de Carthage*
   LARODEC, Tunisia
   rim.faiz@ihec.rnu.tn
3. *Université de Tunis*
   LARODEC, Tunisia
   othmannouha@gmail.com

RÉSUMÉ. *La recherche par mots-clés dans les bases de données relationnelles a suscité un intéet grandissant ces dernières annèes en raison de sa facilité d'utilisation. La plupart de ces recherches non seulement nécessitent un accès préalable aux données, ce qui restreint leur applicabilité si cette condition n'est pas vérifiée, mais aussi renvoient des réponses très génériques. Cependant, fournir aux utilisateurs des réponses personnalisées est devenu plus que jamais nécessaire en raison de la surabondance de données qui peut déranger l'utilisateur. Inspiré par l'application réussie de la technique de filtrage collaboratif dans les systèmes de recommandation, nous proposons une nouvelle approche basée sur les mots-clés pour fournir aux utilisateurs des résultats personnalisés basés sur l'hypothèse que seulement une information sur le schéma de la base de données est fournie.*

ABSTRACT. *Keyword search in relational databases has attracted wide attention over the last years owing to its ease of use. Most recent research not only requires a prior access to the data, which severely restricts their applicability if this condition is not verified, but also returns generic answers. Nevertheless, giving users personalized answers has become more than ever necessary due to the overabundance of data which can be annoying for the user. Inspired by the successful application of the collaborative filtering technique in recommender systems, we propose in this paper a new approach to provide users with customized results based on the hypothesis that only information on the database schema is provided.*

MOTS-CLÉS : *Recherche par mots-clés, systémes de recommandation, Bases de données relationnelles.*

KEYWORDS: *Keyword search, Recommender systems, Relational databases.*

1. Introduction

Over the decades, an explosive amount of structured data has been stored side by side in Relational Databases (RDB). RDB have been widely used owing to the rich information they provide including links between the different entities in the DB. Developing effective query methods for users to easily query huge and complex repositories without the need of technical expertise has become one of the biggest challenges of the database community (Agrawal *et al.*, 2002 ; Aditya *et al.*, 2002).

In fact, the emergence of web search engines has made keyword search the most commonly used search technique. The strength of this latter is that it enables users to easily express their information needs by a few keywords without the need to have knowledge of the database schemas or structured query languages. Nevertheless, such an approach requires a prior access to the data instance in order to build the indices that will pinpoint the different tuples associated with the keywords at run time (Bergamaschi, Domnori *et al.*, 2011). This is a considerable shortcoming since it limits their applicability if a prior access to the data is not possible. Another significant limitation is that the inter-dependencies between the query keywords were ignored. In fact, in reality, a keyword in a query depends on both the relationship between the keyword and the data structure, and the data to which the other keywords in the query are modeled. More concretely, the meaning of each keyword in a user's query also depends on the meaning of the others. On the other hand, with the tremendous development of information technology, the amount of data has been growing exponentially. Thus, finding the desired information in a massive database has become a crucial but also a challenging task. As a matter of fact, the user has to select the appropriate keywords and complete the task of search systems to find the desired answer, which is a tedious task and requires not only an additional effort but also a considerable loss of time. Recommendation Systems are actually a powerful tool to filter data, providing only what the user is most likely looking for. Recently, RSs have been widely used especially in e-commerce websites which take advantage of the personalization ability of these systems to better predict user's intent.

In the present paper, we propose, to the best of our knowledge, the first successful attempt that combine RSs techniques and RDB to overcome the limits of the keyword search in the underexploited context: Keyword search over RDB when no prior access to data is possible. Our novel approach named DBRec aims at returning personalized answers when we have no prior access to DB. The remainder of the paper is structured as follows. Section 1 gives an essential background on keyword search over RDBs and RSs respectively. In Section 2, we review the main existing work on querying RDBs as well as the main methods of RSs. Then, we describe our proposed approach in Section 3. Subsequently, we present in Section 4 our experimental evaluation done to assess the effectiveness of our approach and we report our findings. Towards the end, we conclude and outline our challenges and perspectives.

2. Related work on Querying Relational Databases

Arguably, keyword search has become the standard for seeking information on the Web as it allows the user to easily formulate queries with a few keywords. However, its simplicity comes with a price; keywords are fraught with ambiguity and their intended meaning needs to be explored further (Wang *et al.*, 2008).

Over the years, the Information Retrieval (IR) community has developed advanced approaches for keyword search over documents to return relevant answers to the user. These approaches, though, don't return good results with RDB systems (RDBs), as IR-style search considers tuples as unstructured data, while in RDBs, the retrieved information is spread among tables. Unlike textual documents, the tuples are linked through the foreign-primary key constraints. Thus, foreign-primary key paths connecting tuples that contain keywords, represent an essential ingredient for solving a keyword query over a database. Defining representation models for databases to retrieve these paths is crucial. For these reasons, the direct application of keyword-based approaches to relational databases, where information is fragmented in numerous tables, is neither efficient nor effective (Bergamaschi *et al.*, 2014). Indeed, multiple systems were proposed in the literature, where the most popular ones are BANKS (Aditya *et al.*, 2002), BANKSII (Kacholia *et al.*, 2005), DBXplorer (Agrawal *et al.*, 2002), DISCOVER (Hristidis, Papakonstantinou, 2002), and SQAK (Tata, Lohman, 2008) and the most recent one are KEYMANTIC (Bergamaschi, Domnori *et al.*, 2011), KEYRY (Bergamaschi, Guerra *et al.*, 2011) and SEMINDEX (Chbeir *et al.*, 2014). The objective of these systems is to better cover a keyword query, in order to return answers that matches the user's intent. These approaches can be classified into two broad categories: schema based and tuple-based approaches. The Schema-based approaches model the database schema as a graph, in which the nodes express database relations and edges express interdependence between primary and foreign keys. In these approaches, two phases are needed to build the query answer. First, possible paths are developed from relations that contain a query keyword and the schema graph. Second, fitting queries related to the current tuple are developed following the generated paths. Such approaches can fulfill a keyword query by the use of the schema information to generate SQL queries in RDBs, such as in DBXPLORER, DISCOVER, PRECIS, SQAK, KEYMANTIC, and KEYRY systems. Tuple-based approaches such as BANKS and BANKS II, model the database as a data graph, wherein nodes represent the tuples and edges denote the relationships between a pair of tuples, such as foreign key or primary key dependencies. Unlike schema-based approaches, only one phase is involved for the answer generation, where the schema extraction and the tuple retrieval tasks are mixed. The particularity of a data graph is that nodes and edges are typically weighted, which provides users with more information on how the objects are interconnected.

KEYMANTIC, KEYRY and QUEST (Bergamaschi *et al.*, 2016) tackled the issue of keyword search over RDBs differently; they can provide answers to the user's query without the necessity of a prior access to the data stored in the database to build indices that will locate the tuples. These techniques use intentional knowledge. In other words, they provide results even if we know only the schema of the database. Therefore, KEYMANTIC and KEYRY consider the inter-dependencies of the keywords in the query to compute the best answers. KEYMANTIC computes intrinsic and contextual weights to consider inter-dependencies between keywords and the mappings. For the generation of the top-k best mappings, an extension of the Hungarian algorithm is used. KEYRY is based on the Hidden Markov Models for the generation of the possible configurations, and generalizes the Viterbi algorithm finding the top-K maximum likelihood state sequences.

3. Related work on Recommender Systems

Nowadays, the plethora of available data makes filtering data more than ever necessary to improve the quality of data. In this direction, Recommender Systems (RSs) are designed to personalize users' search, and provide only what might interest them even if they cannot formulate good queries. RSs have been widely used in recent years as they are able to guide users in a personalized way to interesting objects in a large space of possible options (Ricci *et al.*, 2011). Indeed, these systems can predict user's needs and even surprise the user with recommendations that he doesn't know they exist. Basically, RSs can be categorized into three groups: Collaborative Filtering, Content-Based and hybrid approaches.

Collaborative Filtering is probably the most popular and widely used technique in RSs (Ricci *et al.*, 2011). Basically , it recommended information to a user according to the history valuation of all users communally. The main drawback of this technique is the cold-start problem. That is to say, CF based systems may have difficulties to provide recommendations with new items and new users, when no sufficient ratings for items or no enough users are available in the system. Content-Based (CB) recommenders also called Information Filtering, relies on item's description or uses content in the historical search, to guess the potential items that the user might be concerned with. This type of recommender systems, based on content of items rather than on other users' opinions or interactions, use the similarities of the attributes between items. For each item, a profile containing all properties must be created.

A content-based RS was proposed by (Amatriain, Mobasher, 2014) based only on the profile built up by analyzing the content of items rated by the users in the past. The recommendation process basically consists on matching up the attributes of the user profile with the attributes of a content object. Generally, these systems encompass three major components (Ricci *et al.*, 2011): the Content Analyzer which describes the items' content, the Profile Learner which collects data representative of the user preferences and tries to generalize this data in order to construct the user profile, and the Filtering Component which utilizes the user profile to propose apposite items by matching the profile representation against that of items to be recommended. Unlike the CF technique, CB does not suffer from the item cold-start problem, i.e. it has no problem with recommending new and unpopular items. However, the main shortcoming of this technique is that the content analysis often requires domain knowledge and sometimes it is not easy to extract from the items enough relevant information to describe them. Another limitation is that serendipity is hard to obtain with this model. Furthermore, CB recommenders suffer from the overfitting problem when there are not large enough ratings in the training stage. It is worth mentioning that there have been several hybrid recommender systems that combine both CF and CB models to benefit from the characteristics of both and scale down their limitations. For instance, CF systems are more domain-independent than CB systems, while CB can handle better the cold start problem for both new users and new items. However, the aforementioned approaches require a prior access to data to build indices that will locate the tuples related to the given keywords, at run time, except for KEYRY and KEYMANTIC. The necessity of a prior access to data is not fitting reality, since in some cases we cannot access the data contents of a DB such as in e-commerce sites which are subject to recurring updates. To the best of our knowledge, only KEYRY and KEYMANTIC attempt to resolve this issue. KEYRY is based on the Hidden Markov Model, while KEYMANTIC uses essentially multiple similarity measures with the help of some external knowledge. Both techniques mainly exploit terms of the query and the schema information of the DB. Nevertheless, these works are too static since they rely only on similarity distances and some external knowledge which leads to very generic answers and specifically not user-centric responses. In fact, users can query the same information need differently or also may write the same query for different information needs. Obviously, understanding the user preferences is crucial to better address its query and identify what he is really expecting. Thus, we propose a novel approach to overcome the limitation of the keyword search when no prior access to data is possible and deliver personalized and relevant answers to the user's query.

4. Proposed Solution: DBRec

The core idea of our approach, called *DBRec*, is to combine the keyword search over RDB with some techniques used in recommender systems. DBRec is illustrated in Figure 1

and aims at integrating some recommendation and databases concepts to get better personalized answers to a simple keyword-based query posted by a user. It provides recommendations and serendipitous answers even when no prior access to the database is allowed, relying on both schema and users information. The components of the given approach are detailed below.

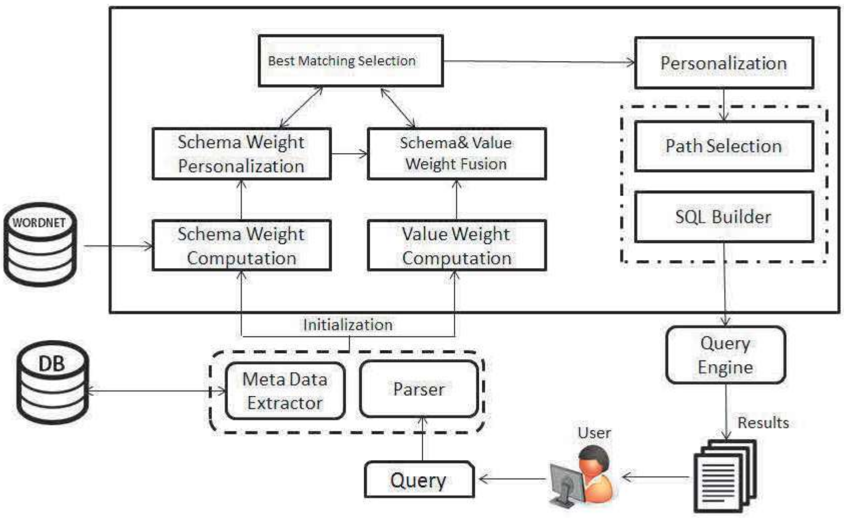

Figure 1. Global architecture of the DBRec Approach

4.1. Schema Terms Matching

Basically, the initialization step includes two phases namely, Schema Weight Computation and Schema Weight Personalization. The meta data extractor extracts the schema information provided with the database to load the first line of the Schema Weight (SW) matrix. The parser separates each term of the query whereby we fill the first column of the given matrix.

4.1.1. Schema Weight Computation

This phase consists in determining which keywords match with schema terms (attributes, relations) starting from a query and the schema information of the database. Attributes and relations are considered as metadata. Different string similarity measures can be used to estimate the keyword-attribute/relation distance such as Levenshtein, Hamming and Cosine distance. Note that Hamming is defined for strings of equal length which is not adequate for our context. We opted for the Levenshtein measure which computes the minimal number of insertions, deletions and replacements needed for transforming a string X into a string Y. However, a simple string similarity between the keyword and the schema term is not enough due to the heterogeneity of the user's vocabulary. For this purpose, we employ WordNet for Word Sense Disambiguation (WSD) to obtain the synonyms, hyponyms and hypernyms related to a given word. Due to the wide available vocabulary, a user may use different words that do not appear in the schema information of the database. Each used keyword is compared to all the synonyms, hyponyms and hypernyms of every schema term to keep the one having the highest similarity.

4.1.2. Schema Weight Personalization

This phase consists in updating the Schema Weight Matrix by making use of the information gathered from the users' profiles. The main idea here is to add the concept of collaborative filtering of the RSs; build profiles for users and use their search history as well as similar users' history to personalize results.

We compute the Personalized Schema Weight (PSW) that uses sessions information to update the SW matrix. Similar sessions are indexed in a table T. Then, we calculate how many times every schema term had the maximum value, for all the queries in the similar sessions. We store the result of this computation in its specific column in the SW matrix. Each value of this column is combined with each one in its analogue column in the SW to get the new PSW values in their corresponding cells. In other words, we weight the values of the SW of each column by a variable that affects the first values depending on the number of times this column had the maximum value.

4.1.3. Best Matching Selection

Computing the best possible matching of keywords to database terms is known as the assignment problem. In fact, we have a number of keywords and a number of schema terms. Any keyword can be assigned to any schema term, incurring some cost that may vary depending on the keyword-schema term assignment. It is required to match all keywords by assigning exactly one keyword to each schema term and exactly one schema term to each keyword in such a way that the total weight of the assignment is maximized. The popular Munkres, a.k.a. Hungarian, algorithm (Munkres, 1957), is a possible solution to this problem but, it provides only the best matching. (Bergamaschi, Domnori *et al.*, 2011) adapted this algorithm to our context, to not stop after the generation of the top one mapping, but continue to generate the other best ones. Besides, the weight matrix is dynamically updated every time a mapping of a keyword to a database term is decided during the calculation.

4.2. *Value Weight Contextualization*

Value Weight Contextualization includes the value weight computation and the schema value weight fusion phases described below.

4.2.1. Value Weight computation

The Value Weight (VW) matrix computation is performed in exactly the same way as in (Bergamaschi, Domnori *et al.*, 2011). The computation is mainly done within the domain information of attributes. KEYMANTIC employed a semantic distance to estimate the relatedness of two concepts. Therefore, every matrix cell in the VW contains a value as an indicator of the eligibility and suitability of the keyword with the attribute domain. The keywords that have already been mapped to schema in the previous step will get assigned 0 in every cell of their lines to ensure that they won't be recomputed in the VW.

4.2.2. Schema Value Weight Fusion

After the computation of the best mapping to schema terms M_i and the value weight matrix, the value weight matrix is updated according the the terms mapped to schema. In keywords queries, a keyword may refer either to a schema term or to a value in a schema term. We contextualize the value weight matrix according to terms mapped as schema terms from the PSW, taking into consideration the keywords positions in the query. The user can use more than one term to describe one concept. We propose the Algorithm 4.2.2 to deal with this issue. This algorithm enables to check for each keyword k if it corresponds to any schema term x from the mapped ones. Then, if k corresponds to a keyword mapped to a schema term x, two situations may arise: If x is a relation R, we add a weight Ω to all the attributes of R for every adjacent keywords A(k) $\cup$ B(k), otherwise, if x is an attribute A of a relation R, we increase the weights of this attribute and the related attributes (with functional dependencies) by Ω for all A(k) $\cup$ B(k) keywords neighboring k.
A(k) and B(k) are two functions that retrieve the following and preceding keywords neighboring k respectively. Ω is a variable, its value is proportional to the distance between keywords. The output of this step is a contextualized Value Weight Matrix $V_j(M_i)$. Again, we will use the extension of the Hungarian Algorithm, this time over $V_j(M_i)$, to get the best assignments.

Algorithm 1 Updated Value Weight Matrix

Input: VW: Value Weight Matrix, Q : Query, M_i : Mapping of the SW
Output: Updated Value Weight Matrix
Compute SVW(VW, Q, M_i) **begin**
  foreach $w \in Q$ **do**
    $x : \exists(w, s) \in M_i$
    if *x* = *null* **then**
      Continue to the next w
    end
    if *x is a Relation R* **then**
      foreach *attribute A of R* **do**
        foreach *w' $\in$ A(w) $\cup$ B(w)* **do**
          SVW[w', R.A] $\leftarrow$ VW[w', R.A] + Ω
        end
      end
    end
    if *x is an attribute A of a relation R* **then**
      foreach *w' $\in$ A(w) $\cup$ B(w)* **do**
        SVW[w', R.A] $\leftarrow$ VW[w', R.A] + Ω
      end
      foreach *attribute A' of R' $\exists$ join path from A to A'* **do**
        SVW[w', R'.A'] $\leftarrow$ VW[w', R.A] + Ω
      end
    end
  end
  return SVW
end

4.3. Generation of the personalized query answers:

The $V_j(M_i)$ with its related M_i is a full matching of the keywords to DB terms, producing together a new combination named A_ij. The score of each combination is the sum of weights of the $V_j(M_i)$ and its associated M_i.

SQL queries can be achieved with the possession of the first-score combinations. Yet, this latter just reside on the keywords match to their adequate database terms, without defining the relations between the terms. Works that process under the same assumption of no prior access to data such as (Bergamaschi, Domnori *et al.*, 2011) only consider the similarity between keywords-attributes/domain of attributes' similarities, which is not always appropriate. To cope with this limitation, we take advantage of the profiles built by our approach, to personalize the answers. Giving a current user making a query in his current session, we compute the similarity of this query with all previous queries in all the sessions using the cosine similarity. The answer that gained interest of the user is credited even if its associated query is fairly similar to the user's current query.

In a nutshell, our contribution consists in adding a Collaborative Filtering aspect to a previous proposed solution for querying databases without a prior access to data. Based on generated profiles, which are in our case a set of sessions created by users, we guarantee a more personalized answer. We try to take into account users' interactions with the system to better predict user's expected answers. The proposed approach ensures more serendipity thanks to the CF that can offer a controllable serendipity.

5. Experimental Evaluation

5.1. Experimental Setup

In our experiments, we used MySQL 5 as the relational database management system and WordNet 3.0 [1] as a lexical database. We explained the content of the Database to no technical users without exposing its schema information. Then, we asked them to propose keywords queries and describe what they expect as answers for their queries. Thereafter, an expert formulates an SQL query for this purpose. We compared the results generated by our approach with those obtained by the expert. We used a fraction of the MovieLens [2] 100K database. In our tests, 18 users were involved, we group each user's queries in a single session. The number of sessions is between 1 and 3 for each user. We used 105 queries distributed among the users sessions. To evaluate DBRec, we were mainly based on the following parameters:

– Number of keywords: As keywords are the main information we are analyzing in our approach, we varied the number of keywords to see its impact on the results.

– Similarity threshold: It principally defines the number of the possible answers. We ranged it from 0.6 to 0.9 and we compare the results with those of KEYMANTIC.

– Number of sessions: As we were based on the Collaborative Filtering aspect, it makes sense to study the impact of sessions on the approach.

Note that for the initialization, we used the Information_schema views provided by MySQL which allows to retrieve metadata about objects in the DB.

5.2. Experimental Results

Figure 2 shows a comparison between KEYMANTIC and DBRec in terms of the percentage of the *1st position answers* which represents the percentage of expected answers by the user generated as the first answers by the system, the percentage of the *not in first position answers* which denotes the expected answers which are not in first positions and the percentage of the *not found answers* which denotes the relevant answers that the system fails to return.

Experiments showed an amelioration in the percentage of the *1st position answers* over the *not in first position answers*. However, the percentages of the *not found answers* for both approaches are quite similar. We run DBRec every time we add a

1. www.wordnet.princeton.edu
2. www.MovieLens.com

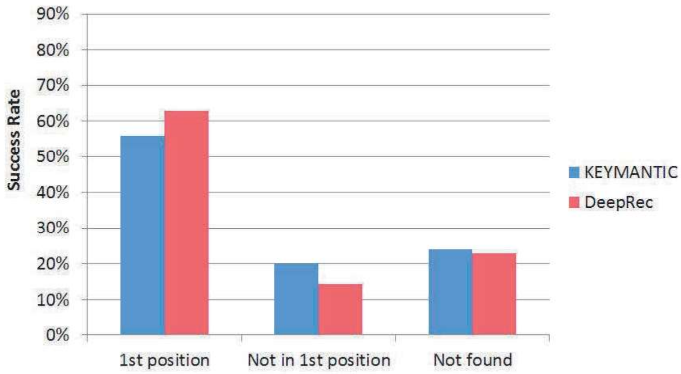

Figure 2. Generated Success Rate for DBRec and KEYMANTIC

session. For the very first sessions, we obtain nearly the same results as KEYMANTIC. This means that even with a cold start, results quality does not decrease. Yet, sessions after sessions some answers have changed, until we obtain the results given in Figure 2. The results of success rate according to the number of sessions are given in Figure 3.

For the keywords number and its impact on the success rate, we computed the percentage of answers that were considered as relevant to the users and returned as the first answer while changing the keywords number in the query as shown in Figure 3.

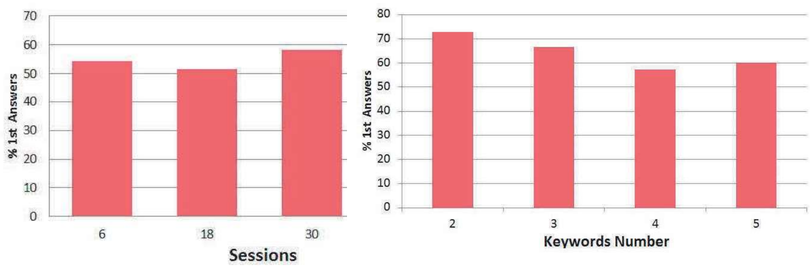

Figure 3. Generated Success Rate according to Sessions and Keywords Number

We remarked that no changes have been noticed for the *not found answers*. Some queries can generate the expected answer, but mostly, not in the first position, except for the ones that have been 'liked' before by the user. The processing time varies from few seconds to minutes, depending on the number of tables related to the queries. For example, the queries that concern one table take around 2 seconds. However, the response time can reach around 2 minutes for the queries including several tables. The execution time needed for both approaches to generate answers is quite similar. The similarity threshold is used to determine the minimum similarity distance required between keywords and terms to be retrieved from the DB. Figure 4 shows how the system reacts when varying the similarity threshold. We can observe the descending allure of this histogram, showing that the larger the value of the threshold is, the less the analyzed terms of DB are and the less the execution time is.

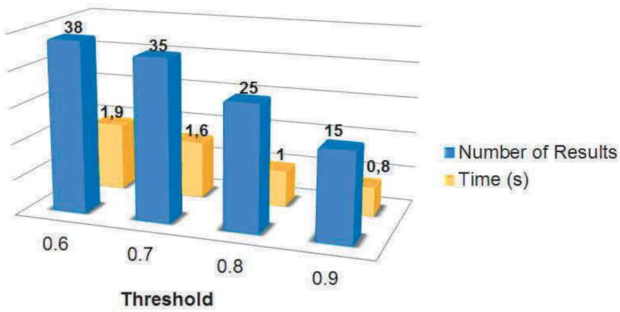

Figure 4. Effect of varying the threshold on DBRec results

5.3. Discussion

Interesting results are presented for users' interactions including sessions and queries. In fact, the number of answers responding to users' expectations has increased. As any collaborative filtering based system, the more users we have and the more interactions they make with the system, the better its performance is. Experiments show that there is no optimal similarity threshold, it depends on the query and users' expectations. By increasing the threshold, both results number and time are reduced. However, by decreasing it, better results are obtained, with a higher serendipity, but it comes with the price of time. Moreover, there isn't an optimal number of keywords. The more words are contained in the query, the longer the time required for results generation. However, the number of keywords may increase either accuracy or serendipity depending on the terms employed by the user. The cold start problem of the Collaborative Filtering technique is resolved by DBRec. This latter allows to have results even with new users, thanks to a back strategy step, insured by Personalized Schema Weight, which maintains the values of the initial matrix obtained before personalization. Furthermore, the personalization step allows to take into account every result that fits the user's request. This is very useful when a user searches an information that has previously sought. However, with KEYMANTIC, regardless the number of times the user interacted with the system, it always recomputes the answers ignoring what the user is probably expecting. The schema quality is a crucial parameter for DBRec. Indeed, having a database containing tables with nearly the same names, structure and attributes data-types may have some limitations. A considerable one is the time needed to generate answers, few minutes for a single query is relatively long, but it is still better than having no result.

6. Conclusion

This paper addressed the problem of processing keyword queries over relational databases under the assumption that no prior access to data is possible. Our contribution consists on providing users with personalized results in this specific context. by extending existing approach with new components and resources. These new components are the personalization of the schema weight matrix and the answers respectively, while the resources are users and information concerning their interactions with the systems. Beyond simply returning generic answers, our findings indicated that our approach called DBRec provides personalized results that better fit the users' intent. Before the generation of the answers, we took advantage of the current user profile to further personalize answers based on the CB technique of RSs; favoring an answer already liked by the user if the similarity between queries are higher than a given threshold. Providing personalized answers when no prior access to data is possible, makes DBRec usable in a large range of applications. In the future, it would be interesting to integrating a semantic-based feature in our approach in order to decrease the percentage of the not found answers.

Bibliography

Aditya B., Bhalotia G., Chakrabarti S., Hulgeri A., Nakhe C., Parag P. *et al.* (2002). Banks: Browsing and keyword searching in relational databases. In *Proceedings of the 28th international conference on very large data bases*, p. 1083–1086.

Agrawal S., Chaudhuri S., Das G. (2002). Dbxplorer: A system for keyword-based search over relational databases. In *Data engineering, 2002. proceedings. 18th international conference on*, p. 5–16.

Amatriain X., Mobasher B. (2014). The recommender problem revisited: Morning tutorial. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, p. 1971–1971.

Bergamaschi S., Domnori E., Guerra F., Trillo Lado R., Velegrakis Y. (2011). Keyword search over relational databases: A metadata approach. In *Proceedings of the 2011 acm sigmod international conference on management of data*, p. 565–576. New York, NY, USA.

Bergamaschi S., Guerra F., Interlandi M., Trillo-Lado R., Velegrakis Y. (2016). Combining user and database perspective for solving keyword queries over relational databases. *Information Systems*, vol. 55, p. 1–19.

Bergamaschi S., Guerra F., Rota S., Velegrakis Y. (2011). A hidden markov model approach to keyword-based search over relational databases. In *Conceptual modeling er 2011*, vol. 6998, p. 411-420.

Bergamaschi S., Guerra F., Simonini G. (2014). Keyword search over relational databases: Issues, approaches and open challenges. In *Bridging between information retrieval and databases*, vol. 8173, p. 54-73.

Chbeir R., Luo Y., Tekli J., Yetongnon K., Ibanez C. R., Traina A. J. *et al.* (2014). Semindex: Semantic-aware inverted index. In *Advances in databases and information systems*, p. 290–307.

Hristidis V., Papakonstantinou Y. (2002). Discover: Keyword search in relational databases. In *Proceedings of the 28th international conference on very large data bases*, p. 670–681.

Kacholia V., Pandit S., Chakrabarti S., Sudarshan S., Desai R., Karambelkar H. (2005). Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on very large data bases*, p. 505–516.

Munkres J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, n° 1, p. 32–38.

Ricci F., Rokach L., Shapira B., Kantor P. B. (2011). *Recommender systems handbook* (vol. 1).

Tata S., Lohman G. M. (2008). Sqak: doing more with keywords. In *Proceedings of the 2008 ACM SIGMOD international conference on management of data*, p. 889–902.

Wang H., Zhang K., Liu Q., Tran T., Yu Y. (2008). Q2semantic: A lightweight keyword interface to semantic search. In *The semantic web: Research and applications*, vol. 5021, p. 584-598.